**“A MESSENGER APLLICATION”**

**PROJECT REPORT**

Submitted for the course: Data Structures And Algorithms (CSE 2003)

By

Vaibhav Khandelwal

Reg. NO. 15BCE0342

Slot: G1

**Name of faculty: Prof. Raghuveer V.R.**



03 May, 2016

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SERIAL NO.** | **TOPIC** | **PAGE NUMBER** |
| 1 | Certificate | 3 |
| 2 | Acknowledgements | 4 |
| 3 | Introduction | 5 |
| 4 | Code | 6-20 |
| 5 | Explanation | 21-23 |
| 6 | Input/output snaps | 24-44 |

**CERTIFICATE**

This is to certify that the project work entitled “**A Messenger Application**” that is being submitted by **Vaibhav Khandelwal (15BCE0342)** for **Data Structures And Algorithms (CSE 2003)** is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

*Place*: Vellore

*Date*: 03 May, 2016

(signature)

**Dr. Raghuveer V.R.**

**Associate Professor,**

**Department of Analytics (SCOPE)**

**ACKNOWLEDGEMENTS**

I gracefully acknowledge the guidance and continuous support given by Prof. Raghuveer V.R. in completing my project successfully. I would also like to thank my Computer Science teacher, Dr. Vijayasherly V. for providing me assistance with various concepts and modules required for my project.

Apart of this, I would also like to thank my parents for supporting me and motivating me in every moment of my life. I also thank my friends and roommates for helping me sort out some bugs in my code and testing it for different cases.

I would also like to thank VIT University Management and Prof. Venkateswaran S  
Dean, School of Computing Science and Engineering for giving me an opportunity to carry out our studies at the University.

Vaibhav Khandelwal

15BCE0342

.

**Introduction**

This project entitled “A Messenger Application” is a very simple and efficient program to implement all the basic features of a messaging app like ‘Facebook.’

This code deals with the following features:

1. **Sign up**: This feature lets one to register on the messenger.
2. **Sign in**: This feature lets the registered users to log in and access all features.
3. **Inbox**: This feature is used to display all the messages of a particular user.
4. **Chat**: This feature lets the user to send messages to any of the registered users.
5. **Friend list**: This option displays all your friends and also the total number of friends in your friend list.
6. **Add friend**: This feature lets one to add other registered users in his/her friend list.
7. **Unfriend**: This is an important feature which lets you to remove any of your friends from your friend list.
8. **Group Message**: This is a very noticeable feature of the app which lets you send the same message to multiple users at the same time.
9. **Log out**: This feature of the app lets a user to end his session and sign out of the app.

**Code:**

//V-Messenger.cpp

//author – Vaibhav khandelwal

#include<iostream>

#include<map>

#include<vector>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

using namespace std;

void page1();

void page2(string);

map<string,string> user;

class frnd

{

vector<string> flist;

struct chat

{

int message\_id;

string sender,receiver;

char mess[50][100];

int num\_of\_message=0;

};

public:

map<string,chat> sms;

void addfrnd(string);

void unfrnd(string);

void dispfrnd(string);

void chat\_on(string,string,char\*);

void inbox(string);

};

void frnd::chat\_on(string uname,string reciepient,char \*message)

{

int flag=0;

sms[reciepient].sender=uname;

sms[reciepient].receiver=reciepient;

map<string,string>::iterator iuser;

for(iuser=user.begin();iuser!=user.end();iuser++)

{

if(iuser->first==reciepient)

{

flag++;

}

}

if(!flag)

{

cout<<"\nSorry!! No such user exist\a\a\n";

page2(uname);

}

int i;

char cuname[20];

for(i=0;uname[i]!='\0';i++)

{

cuname[i]=uname[i];

}

strcpy(sms[reciepient].mess[sms[reciepient].num\_of\_message],cuname);

strcat(sms[reciepient].mess[sms[reciepient].num\_of\_message],": ");

strcat(sms[reciepient].mess[sms[reciepient].num\_of\_message],message);

sms[reciepient].num\_of\_message++;

}

void frnd::inbox(string uname)

{

cout<<"\nWelcome "<<uname<<" to your Inbox\n\n";

cout<<"Messages in your Inbox:\n\n";

for(int i=0;i<=sms[uname].num\_of\_message;i++)

{

puts(sms[uname].mess[i]);

cout<<endl<<endl;

}

}

void frnd::addfrnd(string uname)

{

vector<string>::iterator iv;

map<string,string>::iterator iuser;

int uflag=0,flag=0,flag3=0;

string addname;

cout<<"\nenter the name which you want to add in your friend list"<<endl;

cin>>addname;

cout<<"\nPlease wait! checking for "<<addname<<"......\n";

for(int i=0;i<99999999;)

{

i++;

}

for(iuser=user.begin();iuser!=user.end();iuser++)

{

for(iv=flist.begin();iv!=flist.end();iv++)

{

if(\*iv==addname)

flag3++;

}

if(addname==uname)

{

uflag++;

break;

}

else if(iuser->first==addname)

flag++;

}

if(uflag==1 && flag==0 && flag3==0)

cout<<"\nSorry! Can't add yourself to your friend list\n";

if(flag==1 && uflag==0 && flag3==0)

{

cout<<"\nFound\t"<<addname<<endl;

flist.push\_back(addname);

cout<<"\nCongratulations! "<<addname<<" has been added to your friend list\n"<<endl;

}

else if(uflag==0&&flag==0)

{

cout<<"\nSorry!! this username does not exist\n";

}

else if(flag3)

{

cout<<"\n"<<addname<<" already exists in your friend list\n";

}

}

void frnd::dispfrnd(string uname)

{

int c=0;

vector<string>::iterator i;

i=flist.begin();

cout<<"\nWelcome "<<uname<<" Your Friends are:"<<endl;

for(;i!=flist.end();i++)

{

cout<<"\n"<<\*i<<endl;

c++;

}

cout<<endl<<endl<<"\nTotal friends: "<<c;

}

void frnd::unfrnd(string uname)

{

if(flist.empty())

{

cout<<"\nFriend list empty..!!\a\n";

}

else{

int c=0,pos=0,flag=0;

string name;

cout<<"\nenter the friend name you want to unfriend \n";

cin>>name;

vector<string>::iterator i;

i=flist.begin();

for(;i!=flist.end();i++)

{

c++;

if(\*i==name)

{

flag++;

pos=c-1;

break;

}

}

i=flist.begin();

if(pos || flag)

{

cout<<"\n Removing "<<name<<" from your friend list..\a\a\a\a\n";

flist.erase(i+pos);

cout<<name<<" has been successfully removed from your friend list\n";

}

if(flag==0 && pos==0)

{

cout<<"\nNo such friend exists in your friend list\n";

}

}

}

void page1()

{

int flag=0,choice;

cout<<"\n\t\t\tWELCOME TO THE V-MESSENGER\n";

while(1)

{

cout<<"\nPlease Enter your choice\n 1-Sign\_up\n 2-Sign\_in\n 3-Exit \n \n"<<endl;

cin>>choice;

switch(choice)

{

case 1:

{

string username,pwd;

cout<<"\nchoose a username\n";

cin>>username;

cout<<"\nchoose a password\n";

cin>>pwd;

map<string,string>::iterator iuser;

if(user.empty())

{

user[username]=pwd;

}

else

{

for(iuser=user.begin();iuser!=user.end();iuser++)

{

if(iuser->first==username)

{

cout<<"\nSorry! username not available..please try with some other name\n";

break;

}

else

{user[username]=pwd;

break;}

}

}

break;

}

case 2:

{

string lusername,lpwd;

cout<<"\nenter your username\n";

cin>>lusername;

cout<<"\nenter your password\n";

cin>>lpwd;

map<string,string>::iterator iuser;

for(iuser=user.begin();iuser!=user.end();iuser++)

{

if(iuser->first==lusername && iuser->second==lpwd)

{

cout<<"\nWELCOME "<<lusername<<"!! You have successfully logged in \n";

flag=1;

page2(lusername);

break;

}

}

if(flag==0)

{

cout<<"\nusername or password incorrect\n\t!!Please try again\n";

break;

}

break;

}

case 3:

{

cout<<"\ndisplay\n";

map<string,string>::iterator iuser;

cout<<"\nall users\n";

for(iuser=user.begin();iuser!=user.end();iuser++)

{

cout<<iuser->first<<"\t"<<iuser->second;

}

break;

}

}

if(flag==1)

{

break;

}

}

}

map<string,frnd> f;

void page2(string uname)

{

int choice2,choice3;

main\_menu:

cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;

cout<<"\t\t\t WELCOME "<<uname<<endl<<endl;

cout<<"\n\t \*\*\*\*\*\*\*\*\*\*\*Choose from following menu\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\n1-Inbox\n2-Chat\n3-Friend\_list\n4-Add friend\n5-Unfriend\n6-Group chat\n7-Log out\n";

cin>>choice2;

switch(choice2)

{

case 1:

{

f[uname].inbox(uname);

cout<<"\npress any number except '0' to go back to main menu\n";

cin>>choice3;

if(choice3)

goto main\_menu;

break;

}

case 2:

{

string reciepient;

cout<<"\nPlease enter the reciepient's name:\n";

cin>>reciepient;

char message[100];

cout<<"\nPlease Type Your message :"<<endl;

cin.ignore();

cin.getline(message,100);

f[reciepient].chat\_on(uname,reciepient,message);

cout<<"\nYour message has been sent\n";

cout<<"\npress any number except '0' to go back to main menu\n";

cin>>choice3;

if(choice3)

goto main\_menu;

break;

}

case 3:

{

f[uname].dispfrnd(uname);

cout<<"\npress any number except '0' to go back to main menu\n";

cin>>choice3;

if(choice3)

goto main\_menu;

break;

}

case 4:

{

f[uname].addfrnd(uname);

cout<<"\npress any number except '0' to go back to main menu\n";

cin>>choice3;

if(choice3)

goto main\_menu;

break;

}

case 5:

{

f[uname].unfrnd(uname);

cout<<"\npress any number except '0' to go back to main menu\n";

cin>>choice3;

if(choice3)

goto main\_menu;

break;}

case 6:

{

char ch;

vector<string>rec;

map<string,string>::iterator iuser;

vector<string>::iterator irec;

group:

int flag=0;

string reciepient;

cout<<"Enter reciepient";

cin>>reciepient;

for(iuser=user.begin();iuser!=user.end();iuser++)

{

if(iuser->first==reciepient)

{

flag++;

}

}

if(!flag)

{

cout<<"\nSorry!! No such user exist\a\a\n";

goto group;

}

rec.push\_back(reciepient);

cout<<"Do you want to add more reciepients (y/n)?\n";

cin>>ch;

if(ch=='y')

goto group;

else if(ch=='n')

{

char message[100];

cout<<"\nPlease Type Your message :"<<endl;

cin.ignore();

cin.getline(message,100);

for(irec=rec.begin();irec!=rec.end();irec++)

{

f[\*irec].chat\_on(uname,\*irec,message);

}

}

cout<<"\nYour message has been sent\n";

cout<<"\npress any number except '0' to go back to main menu\n";

cin>>choice3;

if(choice3)

goto main\_menu;

break;

}

case 7:

{

system("cls");

cout<<"\nyou have been successfully logged out\n";

page1();

break;

}

}

}

int main()

{

page1();

return 0;

}

**Explanation**:

The above code implements all the features of a basic messenger which are listed in the introduction using very basic data structure like queue, stacks, etc.

To make the program very simple and efficient and to reduce lines of codes, an extensive use of C++ Standard Template Library (STL) is made. The STL library of C++ lets us to use containers like vectors, maps, lists which fulfil need of basic data type like array to implement queues, stacks etc.

The module wise explanation of the above code is as follows:

**Home Page:** Home page or the first page shows the welcome message along with the three options, i.e. sign up, sign in and exit which are explained below. This is being implemented by calling function ‘void page1()’. It uses switch statement to deal with different cases.

**Sign Up**: This feature makes use of the map container of C++.As soon as this option is selected, the control transfers to the case 1 in switch statement in the page1() function. First It will ask for username then for password. After taking inputs , It inserts both the details in the map taking username as key and password as value.

It also has a feature of checking the username which is already been used. If a match is found it will show invalid username and asks for input again.

**Sign In**: In this feature the input is taken from user and the details are checked in the map. If the match is found it will call the page2() function. Otherwise, it will show a message “NO such user exists”.

**Exit:** By selecting this option you can exit the program. It calls the exit function and terminates the program.

**Class frnd**: This class contains all the functions to implement all the features of page2() function like add-friend(), Display\_frnd(), Chat\_on() etc. It also contains a structure to implement message sending and maintaining inbox.

**Page2()**: As soon as this function is called it displays a menu to use various features as listed in the introduction. This also uses Switch statement to implement various options. Explaination of various cases is stated below:

1. Inbox: This option calls the function, frnd::inbox(string),and displays the array of messages, i.e. mess[i], maintained for a particular user which is currently signed in.
2. Chat: This option calls the function, frnd::chat\_on(string,string),and creates a map of objects, named sms, of structure chat in class frnd and takes reciepient name. In the chat\_on function it equates the sender with the first parameter of chat\_on(string,string) and receiver with the second parameter of chat\_on(string,string). It then check whether the user is registered or not?.

If registered: it inputs the message and copy the sender name in the beginning of the message and the stores this message in the array of messages ,i.e mess[], for that particular reciepient.

If user not registered it shows message “user does not exist”.

1. frnd list: This option calls the function frnd::dispfrnd(string). The function traverses the flist vector of class frnd for that particular user which stores all the friends of that particular user and displays them. Also it increaments the counter and shows its value to show number of friends.
2. Add\_friend: This option calls the function frnd::addfrnd(string). This function first take the input of the name to be added in the friend list then checks it in the map containing all user informations.

If a match is found: it inserts the name at the end of the vector flist for that particular user which is implemented the same way as a queue.

If a match is not found: it shows the message “user does not exist “ and go back to main menu.

It also has features like if you try to add yourself to your friend list ,it will show a message that “you cannot add yourself in your friend list”

Also if a friend is already added , you cannot add him again and again.

1. Unfriend: This option calls the function frnd::unfrnd(string). The function frnd::unfrnd(string) takes the input of the name you want to remove. Then checks for the name in your friendlist, i.e. vector flist.

If match is found: It deletes that element from your queue named flist like we delete any element from a queue.

If match not found: It shows the message “No such user in your friend list”.

Also if your friend list is empty and you perform unfriend , it shows a message “Friend list empty” and get back to the main menu.

1. Group message: This option maintains a different queue of recipients and takes input one by one and inserts them into the queue ‘rec’.

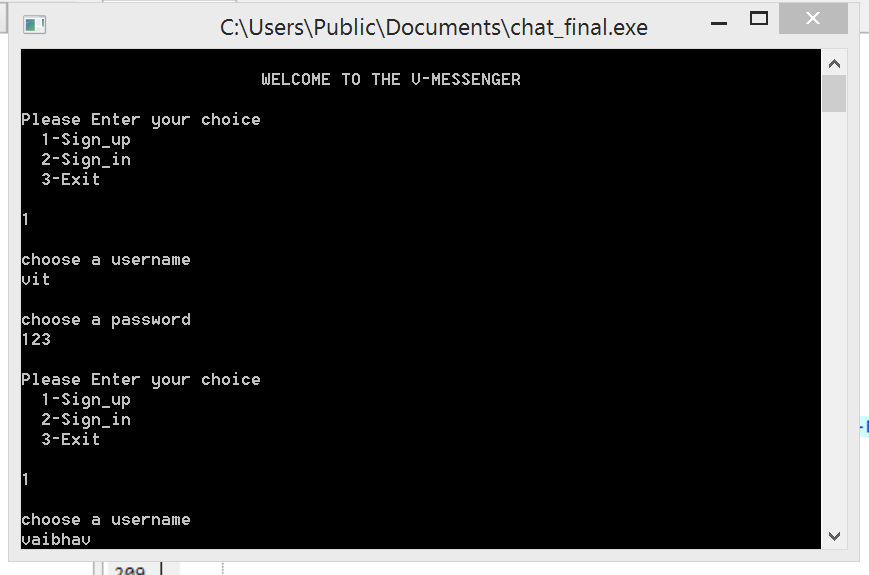
Also it checks every name you enter by choosing option ‘y’, in the map ‘user’ which has all user’s information. If the name is not found in user map then it displays a message “No such user exist.” And takes input for that recipient again.

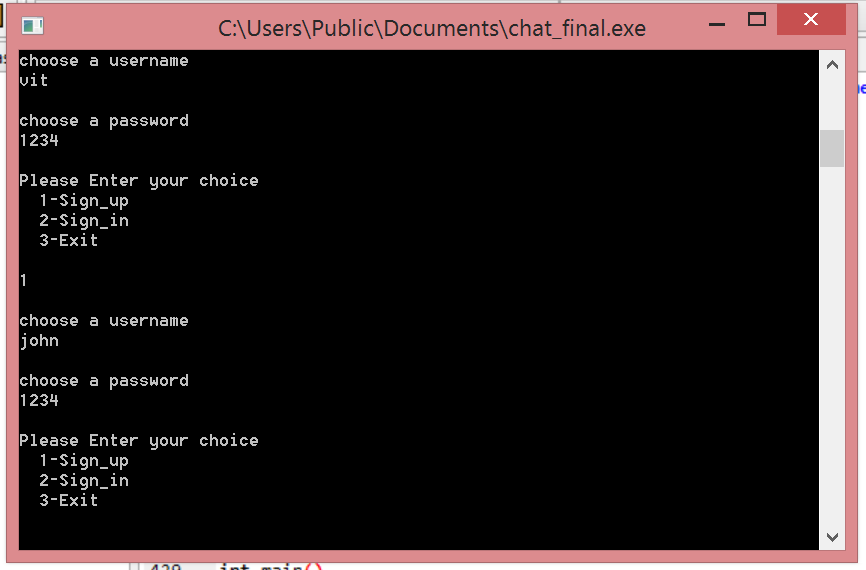
It then takes the message as input and calls the frnd::chat\_on(string,char\*)

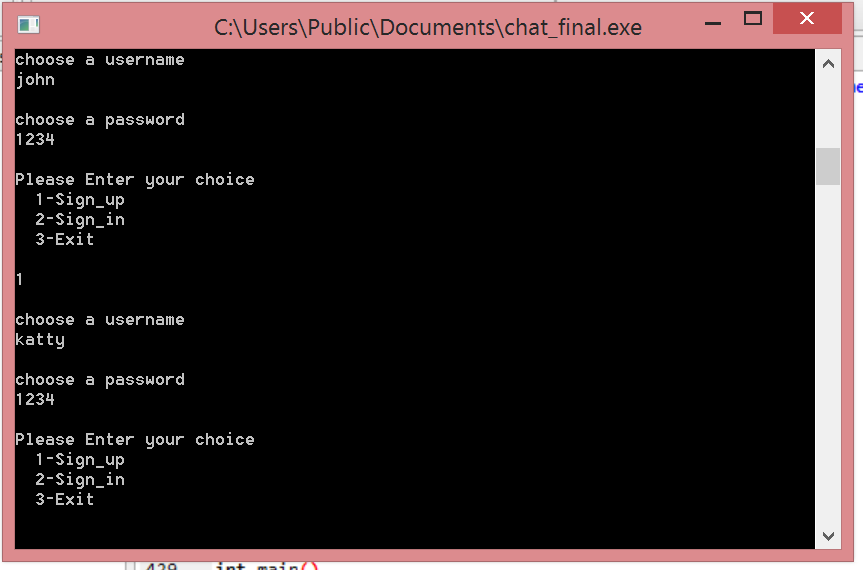
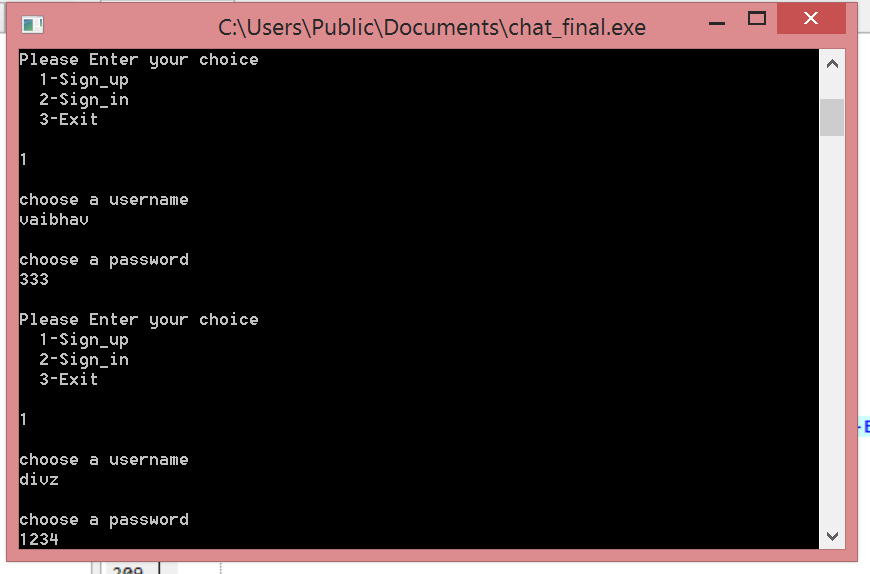
For every recipient individually and thus sends message to each of them.

1. Log out: This option displays the message “You have been successfully logged out” and then calls the page1() function again.

Input/Output Snaps:

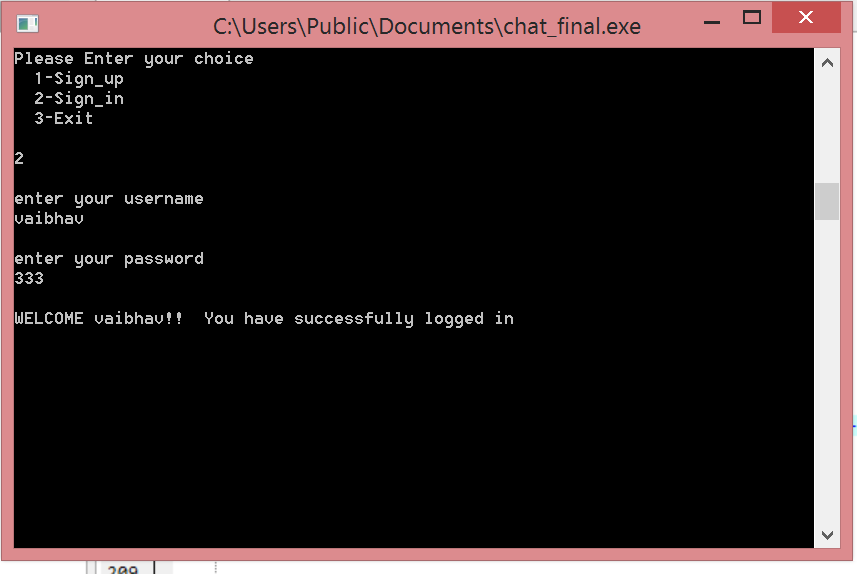


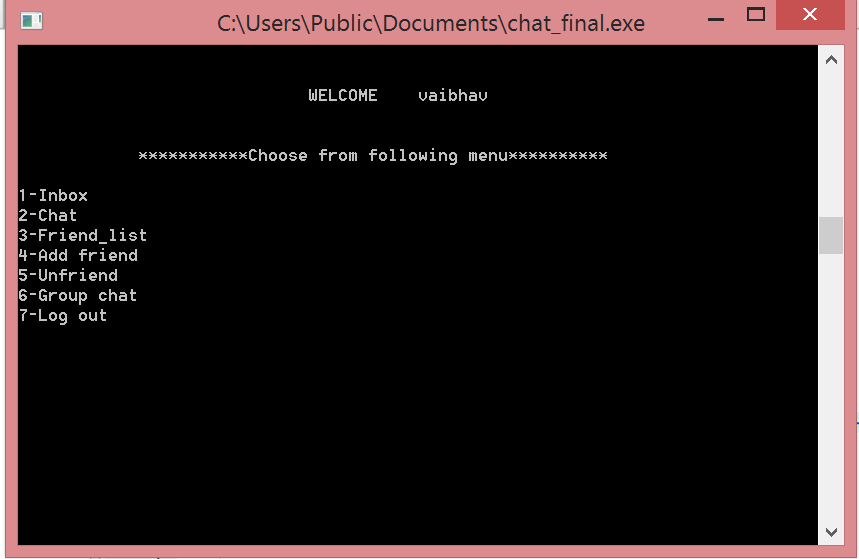


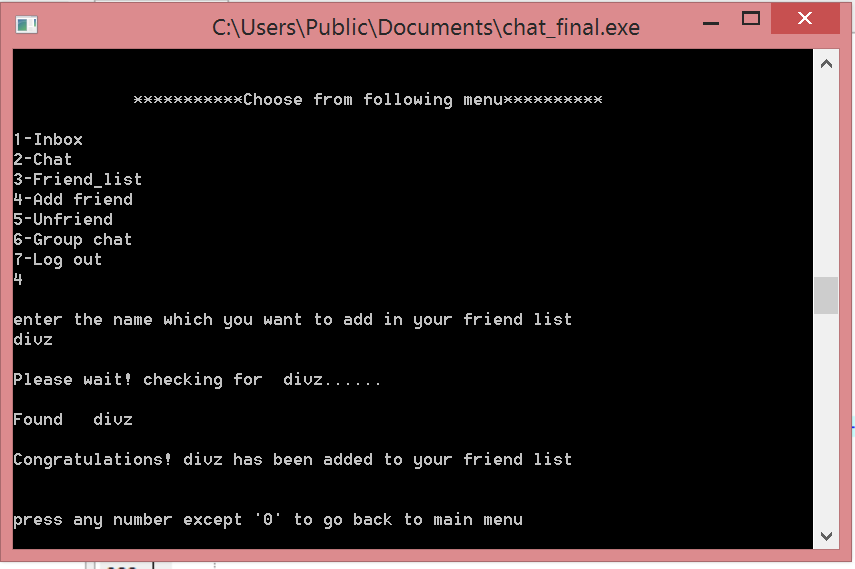


5 users have been signed up: vit,vaibhav,divz,john,katty.

Vaibhav signs in now:



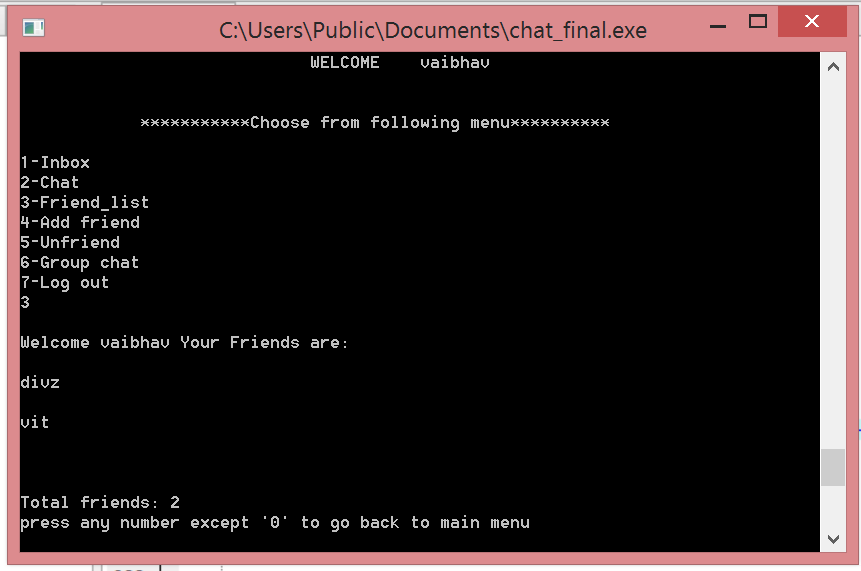




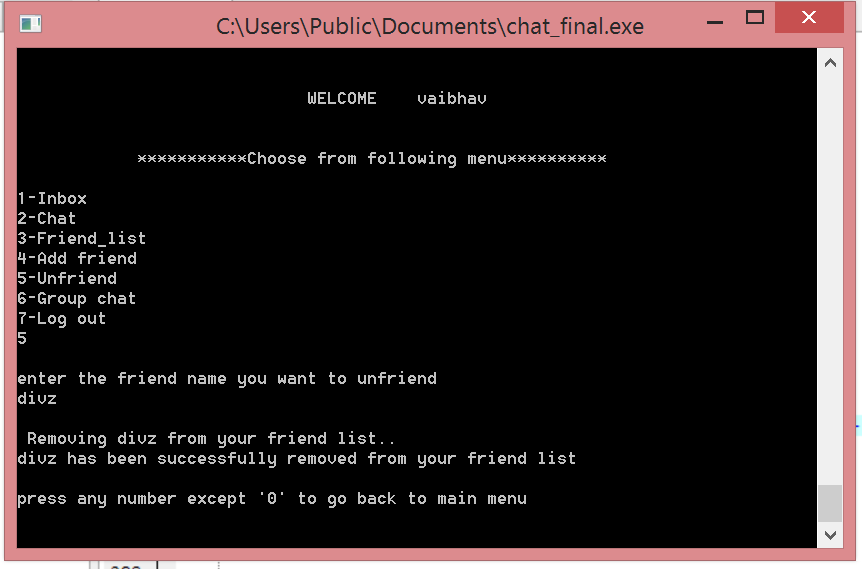
Vaibhav added divz to his friend list

Also added vit to his friend list similiarly.

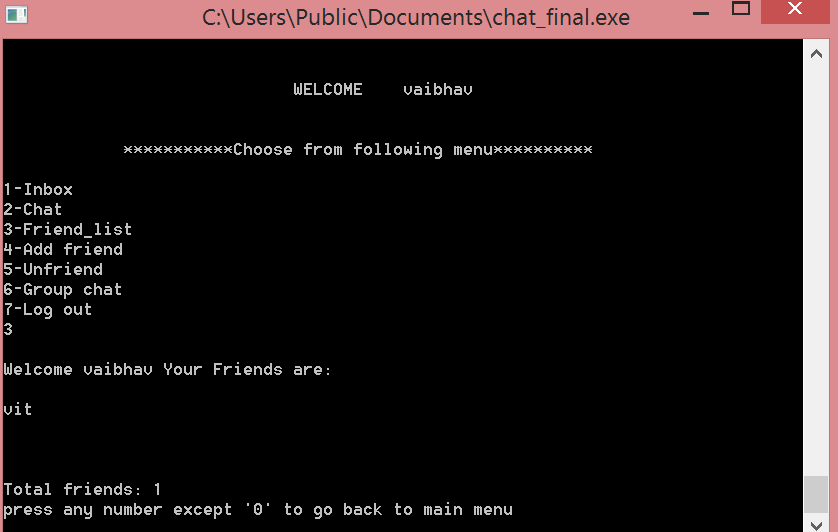
Now friend list after both added:



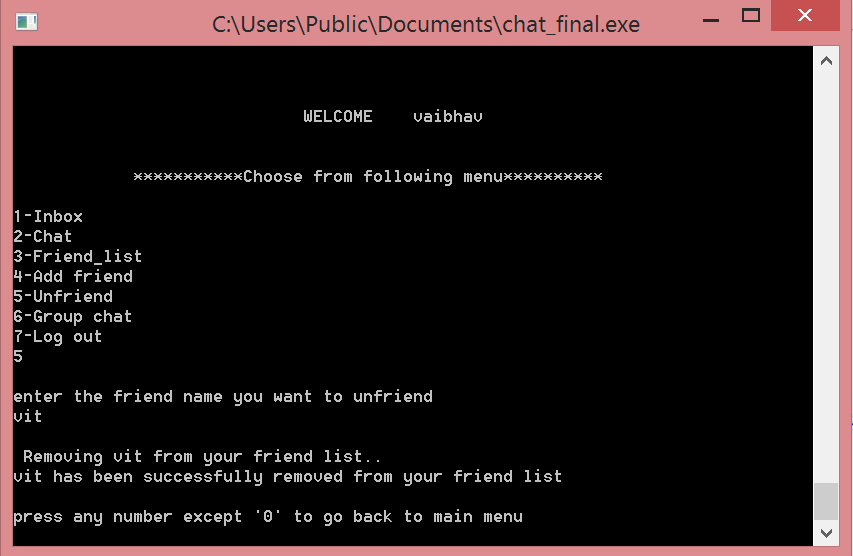
Vaibhav unfriends divz:



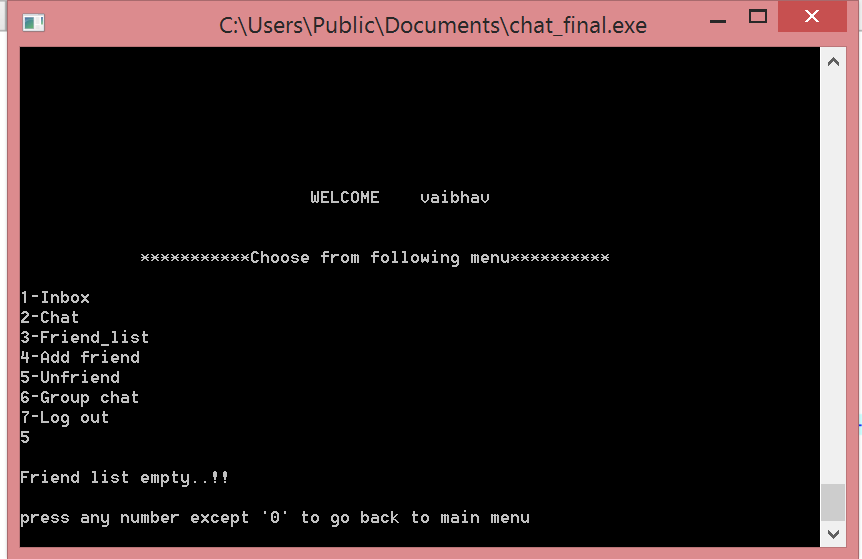
Friend list after unfriend divz:



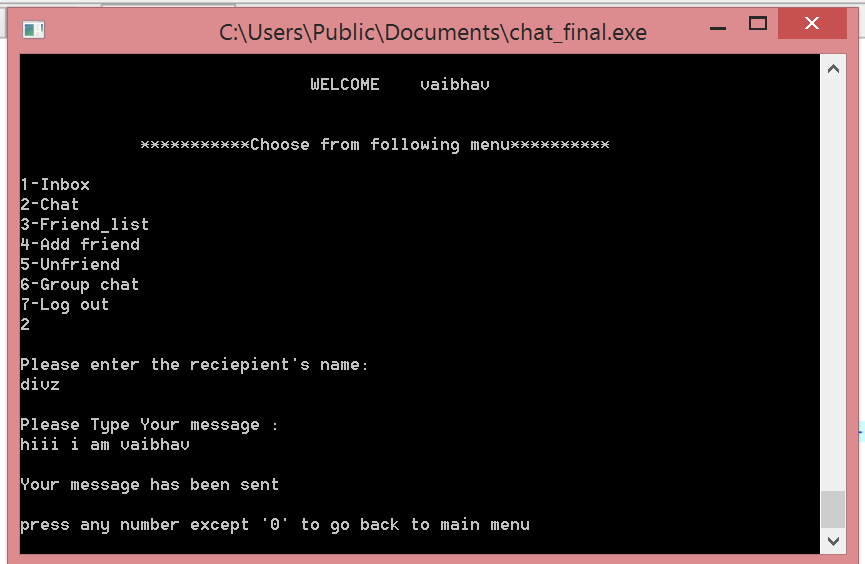
Vaibhav unfriends vit:



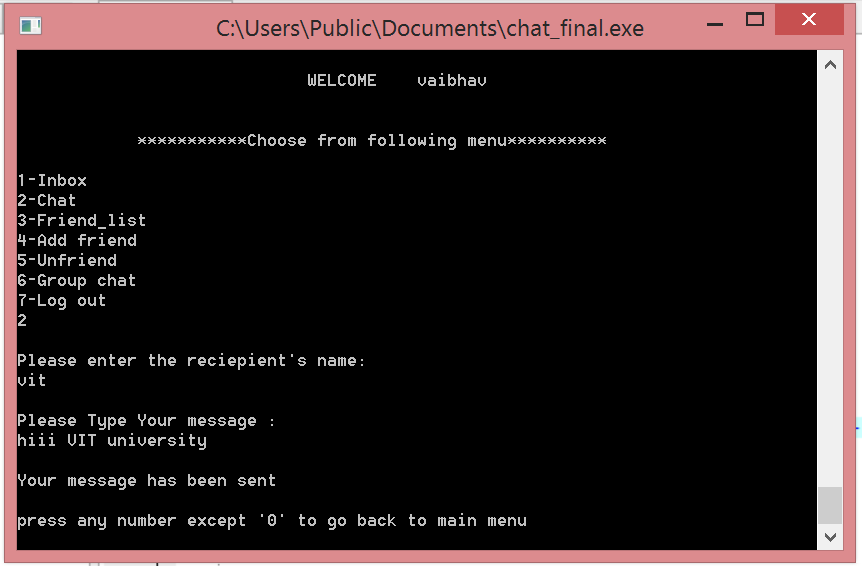
Unfriend after empty friendlist.



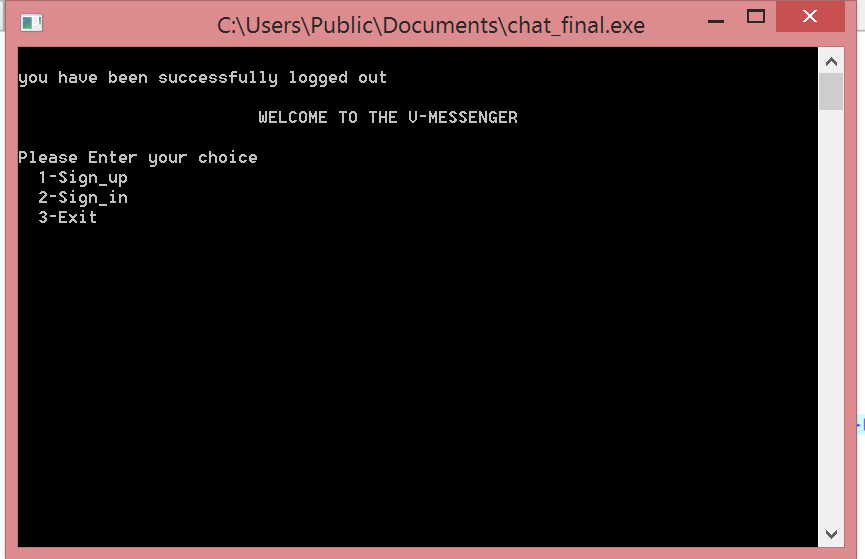
Vaibhav sends message to divz:



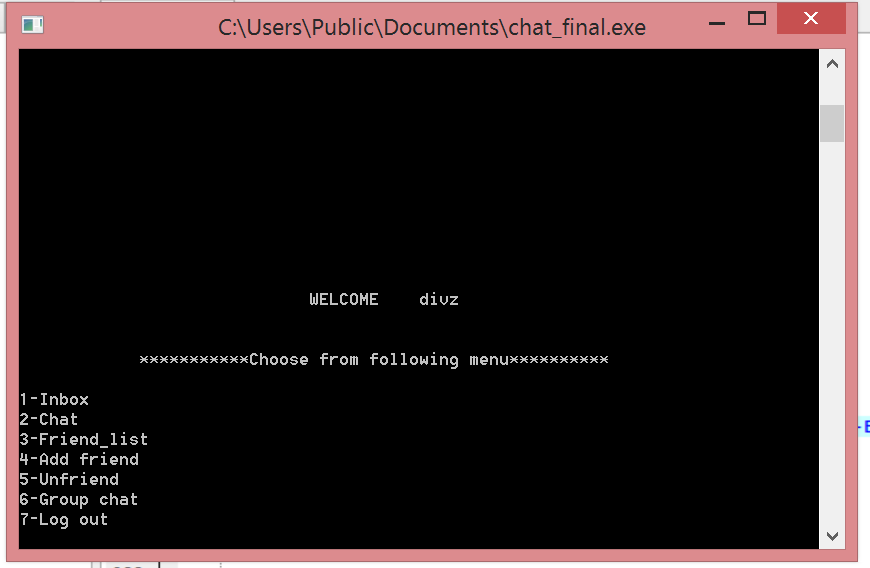
Vaibhav sends message to vit:

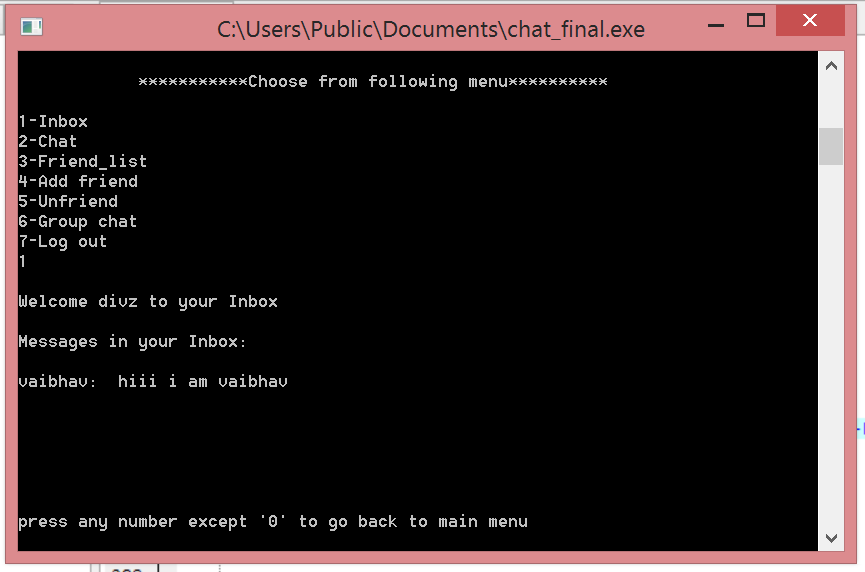


Vaibhav signs out:

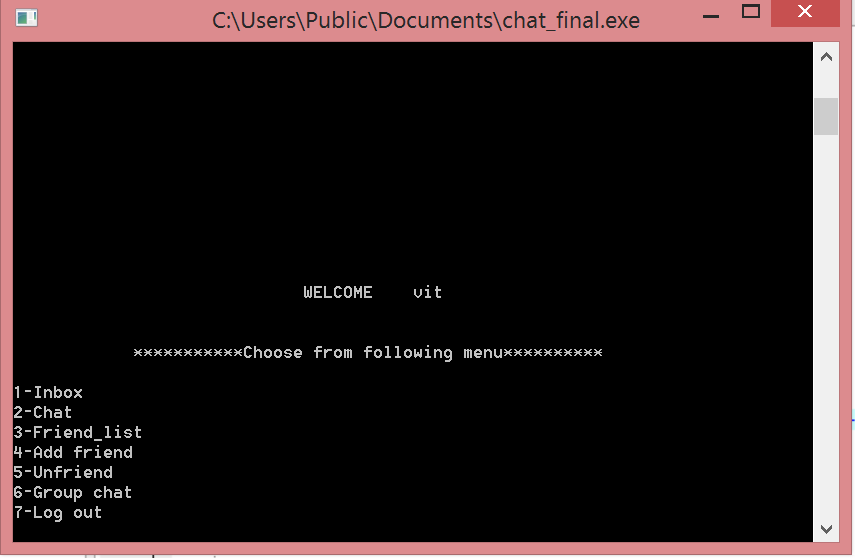


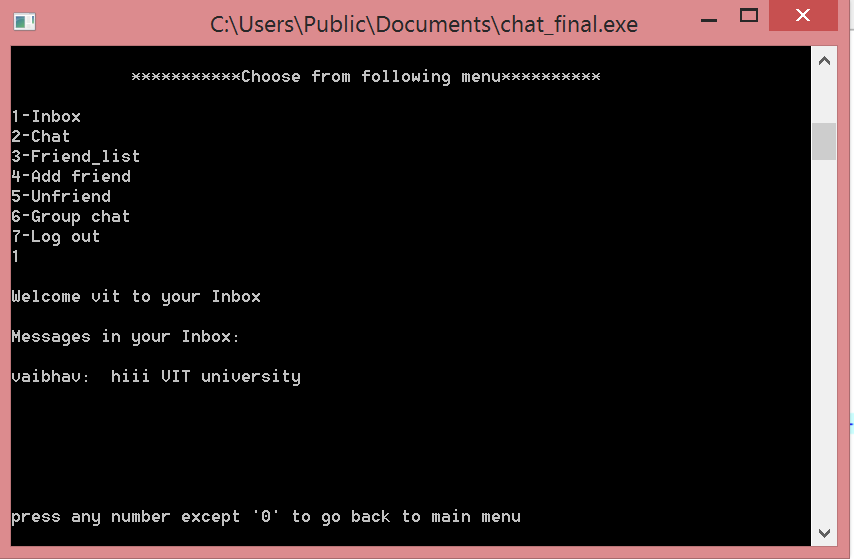
Divz sign in:



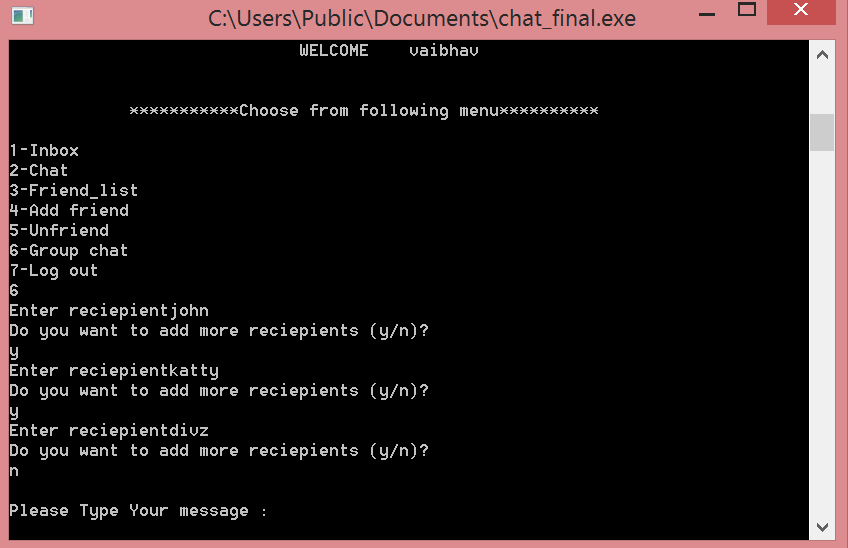
Divz inbox: 

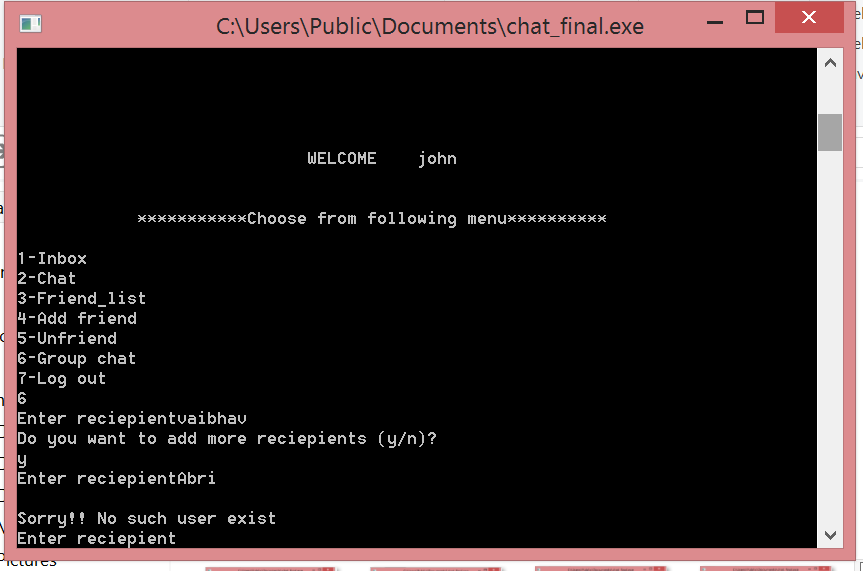
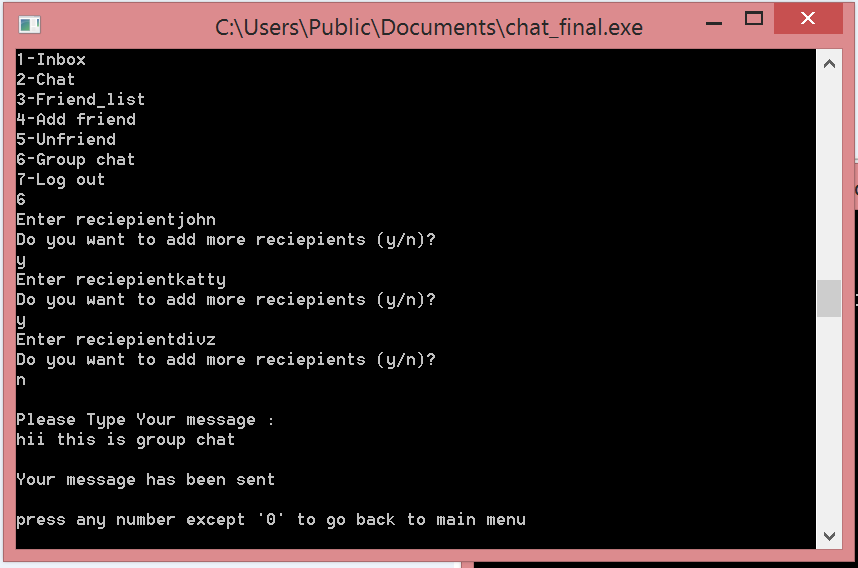
Divz log out and then VIT sign in:

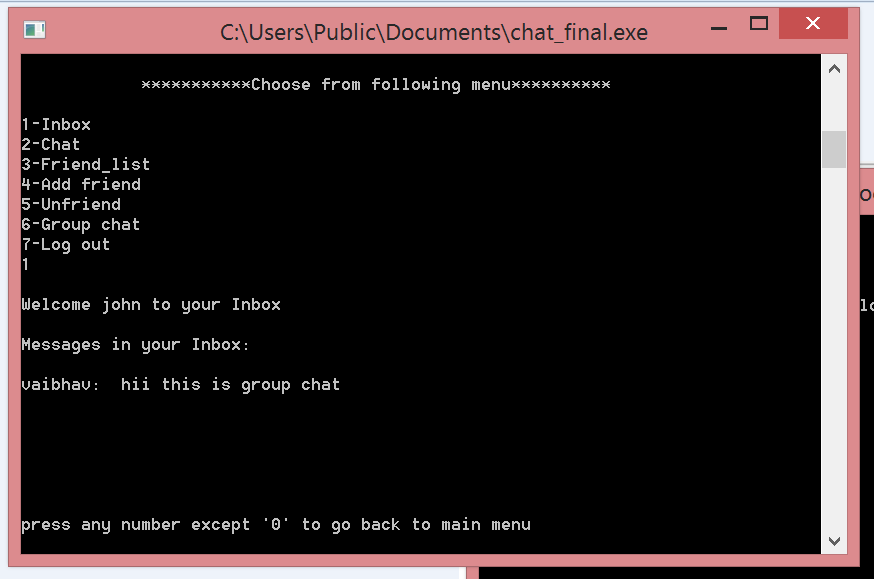


Vit inbox: 

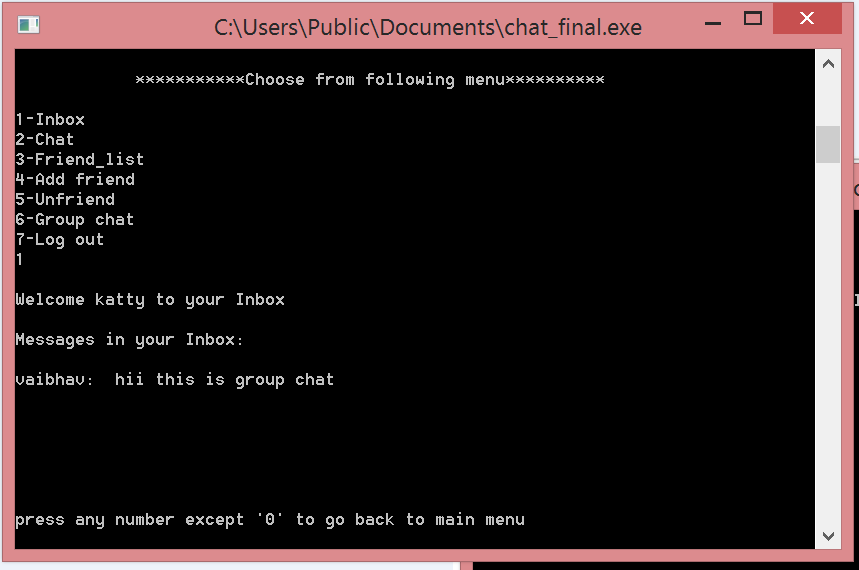
Vit logs out then vaibhav sign in again and open group message and add reciepents:



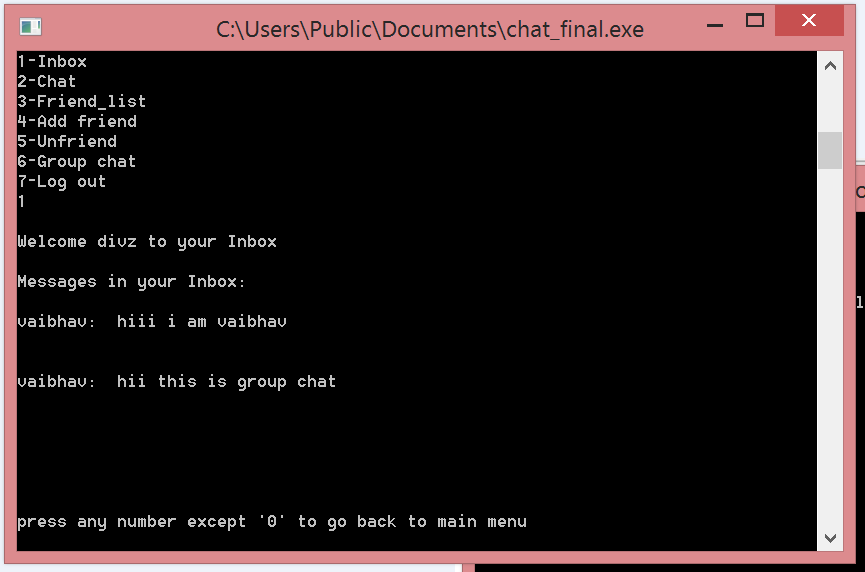
If wrong reciepent added:Now vaibhav sends message to divz john and katty

Now vaibhav log out and john sign in, opens his inbox:

John sign out and katty log in and open her inbox:

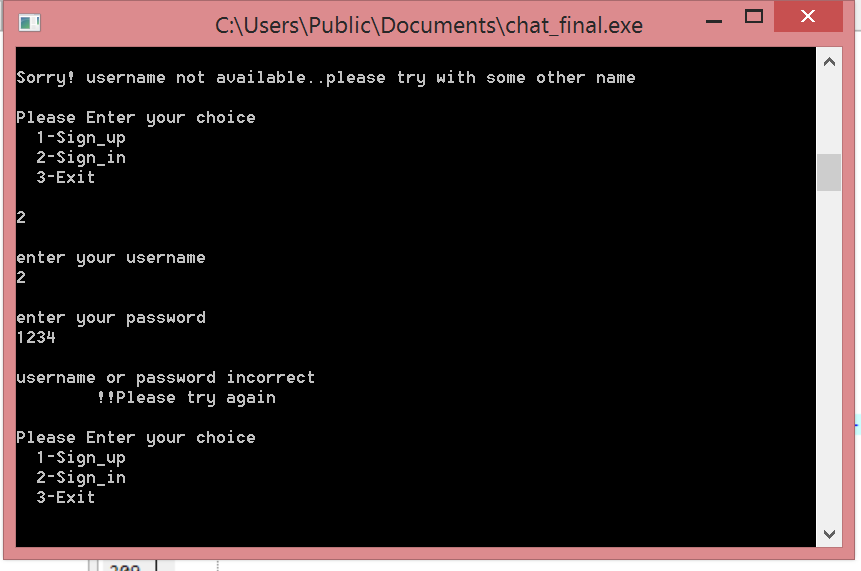


Katty log out and divz sign in. Her inbox contains both messages previously sent by vaibhav and now in group chat:

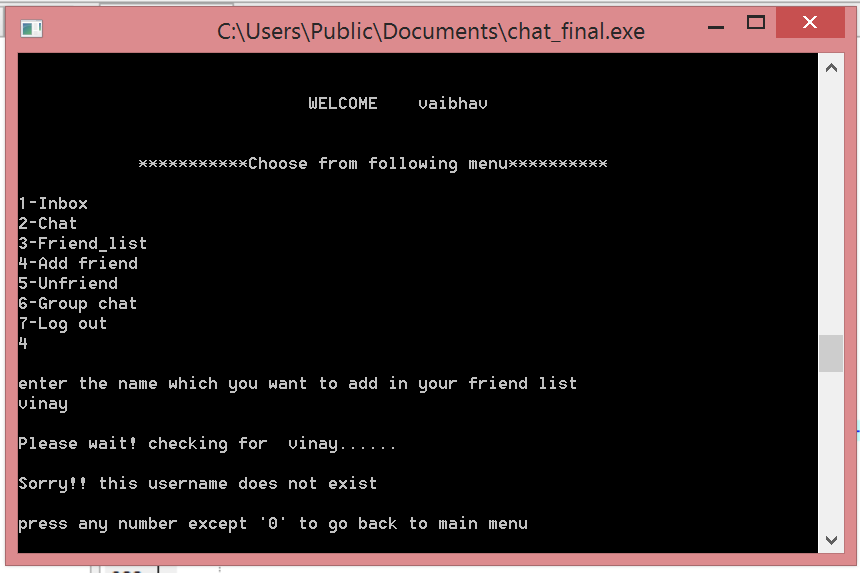


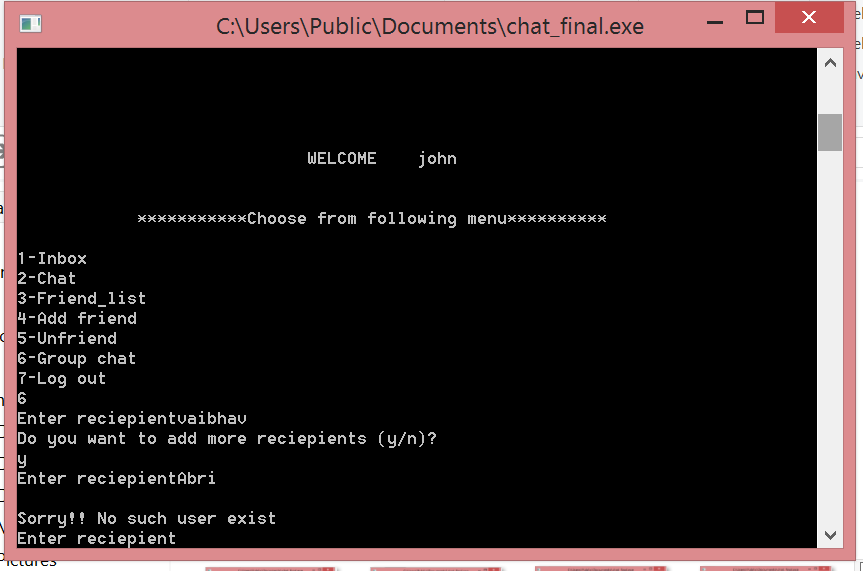
Some Wrong Cases:

1.wrong sign in:



2.wrong add friend:



3.group chat wrong reciepient:

\*\*THE END\*\*